REMARKS

In the Official Action mailed on **8 March 2007**, the Examiner reviewed claims 1-4, 6-16, and 18-25. Claims 1-4, 6-16 and 18-25 were provisionally rejected on the ground of non-statutory obviousness-type double patenting as being unpatentable over claim 1, 3, 5-13, 15, and 17-25 of co-pending Application No. 10/637,166. Claims 1-4, 6-16, and 18-25 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Moss et al (*Transactional Memory: Architectural Support for Lock-Free Data Structures*, hereinafter "Moss"), in view of Oplinger et al (*Enhancing Software Reliability with Speculative Threads*, hereinafter "Oplinger"), further in view of Rajwar et al. (*Speculative Lock Elison: Enabling Highly Concurrent Multithread Execution* , hereinafter "Rajwar").

Obviousness-Type Double Patenting Rejection

The Examiner provisionally rejected claims 1-4, 6-16, and 18-25 on the ground of non-statutory obviousness-type double patenting as being unpatentable over claims 1, 3, 5-13, and 17-25 of co-pending Application No. 10/637,166 in view of Rajwar and Moss. More specifically, Examiner rejected the claims because the claims refer to acquiring a lock after unsuccessfully retrying transactional execution several times, in order to guarantee forward progress of the code.

Applicant has amended claim 1, 13, and 25, removing the references to the acquisition of a lock after retrying transactional execution, rendering the double patenting rejection moot. Hence, Applicant respectfully requests the withdrawal of the double patenting rejection.

Rejections under 35 U.S.C. § 103(a)

Examiner rejected claims 1-4, 6-16, and 18-25 under 35 U.S.C. § 103(a) as being unpatentable over Moss in view of Oplinger, further in view of Rajwar. Examiner argues:

> "Oplinger teaches a transactional programming model in which his abort/fail instruction not only eliminates the speculative state, but also branches the program to an address that was previously set by a TRY instruction (Section 3.2). The advantage of having an instruction is being able to go to an error handling case in the event of an abort (see Section 3.2, the second code example, where the system jumps to an error message on an abort), giving the programmer more control over the execution and troubleshooting of his program (see Office Action, page 5)."

Applicant respectfully points out that Moss and Oplinger are fundamentally distinct from embodiments of the present invention because the combination of Moss and Oplinger discloses an ABORT instruction that is limited to aborting transactional execution and branching to a location set by the TRY instruction.

Moss discloses a very simple version of an ABORT instruction that is limited to terminating transactional execution and discarding "all updates to the write set" (see Moss, section 2.1). On the other hand, Oplinger discloses an ABORT instruction that is confined to discarding the speculative state and setting the program counter to the address specified by the last TRY instruction (see Oplinger, section 3.2). Hence, in the broadest reading, the combination of Moss and Oplinger discloses an ABORT instruction with the function disclosed by Oplinger.

In contrast, embodiments of the present invention provide a FAIL instruction with multiple functions that provides a programmer much more control over program execution. Using the FAIL instruction, the programmer can:

> (1) terminate transactional execution and **branch to a branch target** specified by a corresponding start transactional execution (STE) instruction (see par. [0092] of the instant application);

(2) terminate transactional execution and **branch to an alternative branch target** specified in the FAIL instruction (see par. [0092] of the instant application); or

(3) **update the state information of the processor** with information about the failure, and then to continue transactional execution, wherein **the processor handles the error at a later time** (see par. [0093] of the instant application).

Applicant avers that the FAIL instruction is distinct from Oplinger's ABORT instruction because the FAIL instruction can **operate separately from, as well as in combination with, the STE instruction**. In other words, the STE instruction can provide one branch target that is available to be used by each FAIL instruction during transactional execution. In addition, each FAIL instruction can provide an alternative branch target that leads to a specific error-handling routine. Thus, all FAIL instructions in a given transactional block need not use the same branch target (e.g., the branch target included with the STE instruction). For example, a programmer can use the STE branch target as a generic failure case, while providing alternative branch targets for specific failure cases in local FAIL instructions (see par. [0092] of the instant application). Alternatively, the STE instruction can provide no branch target and each FAIL instruction can provide a branch target.

Applicant also avers that the FAIL instruction is distinct from Oplinger's ABORT instruction because **the FAIL instruction provides the programmer with the ability to delay the processing of failures in certain cases**. In other words, the FAIL instruction can be used to record the occurrence of a condition which the processor can handle later (i.e., which does not require immediate handling). For example, if one or more operations are to be performed before the processor leaves transactional execution, the FAIL instruction can set a flag. The processor can then perform the operations before handling a subsequent commit instruction (see par. [0093] of the instant application).

Moss and Oplinger disclose an ABORT instruction that is confined to aborting transactional execution and branching to a location set by the TRY instruction. Nothing in Moss or Oplinger, alone or in concert suggests a FAIL instruction that provides an alternative branch target or a FAIL instruction which sets state information within the processor to indicate that a failure has occurred.

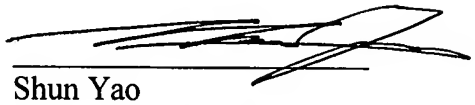
Applicant has amended claims 1, 13, and 25 to clarify that the FAIL instruction can provide an alternative branch target or set state information within the processor to indicate that a failure has occurred. The amendments find support in par. [0092] and [0093] of the instant application. Applicant has also cancelled claims 4 and 16. No new matter has been added.

Hence, Applicant respectfully submits that independent claims 1, 13, and 25 as presently amended are in condition for allowance. Applicant also submits that claims 2-3 and 6-12, which depend upon claim 1, and claims 14-15 and 18-24, which depend upon claim 13, are in condition for allowance for reasons of the unique combinations recited in such claims.

CONCLUSION

It is submitted that the present application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By 

Shun Yao
Registration No. 59,242

Date: 25 April 2007

Shun Yao
Park, Vaughan & Fleming LLP
2820 Fifth Street
Davis, CA 95618-7759
Tel: (530) 759-1667
FAX: (530) 759-1665
Email: shun@parklegal.com